

# LEARNING OF STATE-SPACE MODELS WITH HIGHLY INFORMATIVE OBSERVATIONS: A TEMPERED SEQUENTIAL MONTE CARLO SOLUTION

ANDREAS SVENSSON, THOMAS B. SCHÖN, FREDRIK LINDSTEN

*Department of Information Technology  
Uppsala University, Sweden*

**ABSTRACT.** Probabilistic (or Bayesian) modeling and learning offers interesting possibilities for systematic representation of uncertainty based on probability theory. Recent advances in Monte Carlo based methods have made previously intractable problem possible to solve using only the computational power available in a standard personal computer. For probabilistic learning of unknown parameters in nonlinear state-space models, methods based on the particle filter have proven useful. However, a notoriously challenging problem occurs when the observations are highly informative, i.e. when there is very little or no measurement noise present. The particle filter will then struggle in estimating one of the basic component in most parameter learning algorithms, the likelihood  $p(\text{data}|\text{parameters})$ . To this end we suggest an algorithm which initially assumes that there is artificial measurement noise present. The variance of this noise is sequentially decreased in an adaptive fashion such that we in the end recover the original problem or possibly a very close approximation of it. Computationally the parameters are learned using a sequential Monte Carlo (SMC) sampler, which gives our proposed method a clear resemblance to the SMC<sup>2</sup> method. Another natural link is also made to the ideas underlying the so-called approximate Bayesian computation (ABC). We provide a theoretical justification (implying convergence results) for the suggested approach. We also illustrate it with numerical examples, and in particular show promising results for a challenging Wiener-Hammerstein benchmark.

## 1. INTRODUCTION

Probabilistic (or Bayesian) modeling and learning offers interesting and promising possibilities for a coherent and systematic description of model and parameter *uncertainty* based on probability theory [30, 33]. The computational tools for probabilistic learning in state-space models have lately been developed. In this paper, we study probabilistic learning based on measured data  $\{y_1, \dots, y_T\} \triangleq y_{1:T}$ , which we assume to be well described by a nonlinear state-space model with (almost) no measurement noise

$$(1a) \quad p(x_t | x_{1:t-1}, \theta) = f(x_t | x_{t-1}, u_{t-1}, \theta),$$

$$(1b) \quad y_t = g(x_t),$$

with some unknown parameters  $\theta \in \Theta$  which we want to learn. The lack of measurement noise in (1b) gives a deterministic mapping  $g : X \mapsto Y$  from the unobserved states  $x_t \in X$  to the output  $y_t \in Y$ , on the contrary to (1a) which describes uncertainty about  $x_t$ , mathematically encoded as a probability density  $f$  over  $x_t$  conditional on  $x_{t-1}$  and possibly an exogenous input  $u_{t-1}$ . We refer to this uncertainty as *process noise*, but its origin does not have to be a physical noise, but possibly also due to. The reasoning and contributions of this paper will be applicable also to the case where the relationship (1b) contains uncertainty, *measurement noise*, if its variance is much smaller than the process noise. As a general term, we refer to the model as having *highly informative observations*. Furthermore,  $g$  could also be allowed to depend on  $\theta$  and  $u_t$ , but we omit that possibility for notational clarity.

Models on the form (1) may arise in several practical situations, for instance in a mechanical system where the measurements can be made with good precision but some unobserved forces are acting on the system. The situation may also appear if, yet again, the measurements can be made with good precision, but the user's understanding of the physical system is limited, which in the probabilistic framework also is to be modeled as an uncertainty in  $f$ .

---

*E-mail address:* andreas.svensson@it.uu.se, thomas.schon@it.uu.se, fredrik.lindsten@it.uu.se.

The model (1) defines, together with priors on  $\theta$ , a joint probabilistic model  $p(y_{1:T}, x_{1:T}, \theta)$ . Probabilistic learning of the parameters  $\theta$  amounts to computing the parameter posterior  $p(\theta | y_{1:T})$ , where we have conditioned the probabilistic model on data  $y_{1:T}$  and marginalized over all possible states  $x_{1:T}$  (we omit the known  $u_{1:T}$  to ease the notation). Although conceptually clear, the computations needed are typically challenging, and almost no cases exist that admit closed-form expressions for  $p(\theta | y_{1:T})$ . Methods based on particle filters have proven useful to compute  $p(\theta | y_{1:T})$  in nonlinear state-space models, as outlined in the accompanying paper [37] (see also, e.g., [36, 24]). The idea in the particle filter and its related methods is to represent distributions of interest, such as the posterior  $p(\theta | y_{1:T})$ , with Monte Carlo samples. The samples can later be used to estimate functions of the parameters  $\theta$ , such as their mean, variance, etc., as well as making predictions of future outputs  $y_{T+1}$  etc. In probabilistic learning methods, the particle filter is often used as a building-block for handling the unknown states  $x_t$ , and in particular to compute an estimate  $z$  of the likelihood

$$(2) \quad p(y_{1:T} | \theta) = \int p(y_{1:T}, x_{1:T} | \theta) dx_{1:T},$$

which is a central object in most probabilistic learning algorithms. The peculiarity in the problem studied in this paper is the (relative) absence of measurement noise in (1). This seemingly innocent detail is a show-stopper for the standard algorithms based on the particle filter, since the quality of the likelihood estimate  $z$  tends to be very poor if the observations are highly informative in the model.

This problem has a connection to the literature on approximate Bayesian computations (ABC, [3]), where some observations  $y$  are available, as well as a model (not necessarily a state-space model) with some unknown parameters  $\theta$ . In ABC problems, however, the model is only capable of *simulating* new synthetic observations  $\hat{y}(\theta)$  and the likelihood  $p(y | \theta)$  cannot be evaluated. In problems of this form, the ABC idea is to construct a distance measure between the real observations  $y$  and the simulated synthetic observations  $\hat{y}(\theta)$ , and take this measure (which becomes a function of  $y$  and  $\theta$ ) as a substitute for  $p(y | \theta)$ .

In this paper, we propose a novel algorithm for the purpose of learning  $\theta$  in (1). Our idea is to start the algorithm by assuming that there *is* a substantial amount of measurement noise which removes the computational problem with  $z$ , and then gradually decrease the imagined measurement noise variance simultaneously as the parameters  $\theta$  are learned. The assumption of imagined measurement noise resembles the ABC methodology. The sequence of gradually decreasing measurement noise variance can be seen as tempering, which we will combine with a sequential Monte Carlo sampler [14] to obtain an algorithm with theoretical guarantees which generates samples from the posterior  $p(\theta | y_{1:T})$ .

## 2. BACKGROUNDS ON PARTICLE FILTERING AND TEMPERING

In this section we will provide some background on the Monte Carlo methods particle filters, Markov chain Monte Carlo (MCMC) and related methods. For a more elaborate introduction, please refer to, e.g., [37, 10, 34]. We will in particular discuss why models on the form (1) is problematic for most existing methods, and also discuss the idea of tempering.

**2.1. Particle filtering, PMCMC and SMC<sup>2</sup>.** The bootstrap particle filter was presented in the early 1990's [20, 17] as a solution to the state filtering problem (computing  $p(x_t | y_{1:t})$ ) in nonlinear state-space models. The idea is to propagate a set of  $N_x$  Monte Carlo samples  $\{x_t^n\}_{n=1}^{N_x}$  along the time dimension  $t = 1, 2, \dots, T$ , and for each  $t$  the algorithm follows a 3-stage schema with resampling (sampling ancestor indices  $a_t^n$  based on weights  $w_{t-1}^n$ ), propagation (sampling  $x_t^n$  from  $x_{t-1}^{a_t^n}$  using (1a)) and weighting (evaluate the ‘usefulness’ of  $x_t^n$  using (1b) and assigning it to the weight  $w_t^n$ ). The algorithm will be given as Algorithm 2, and a more elaborate introduction can be found in [37]. The samples are often referred to as particles, and provide an empirical approximation  $\hat{p}(x_t | y_{1:t}) = \sum_{n=1}^{N_x} \delta_{x_t^n}(x_t)$  (with  $\delta$  the Dirac function) of the filtering distribution  $p(x_t | y_{1:t})$ . Since the particle filter itself builds on Monte Carlo ideas, the outcome of the algorithm will be different every time the algorithm is run.

*The particle filter algorithm is only applicable when the state-space model does not contain any unknown parameters.* It has, however, been realized that the particle filter does not only solve the filtering problem, but can also be used to estimate the likelihood  $p(y_{1:T} | \theta)$  of a state-space model by using the empirical approximation  $\hat{p}(x_t | y_{1:t})$  in (2) and hence approximate the integral with a sum. We will denote the obtained estimate with  $z$ , and it can be shown [32] that  $z$  is in fact an unbiased estimator of the likelihood,  $\mathbb{E}[z] = p(y_{1:T} | \theta)$ . With unbiased we mean that the average of the

estimate  $z$  if the particle filter algorithm is run many times (for the same model, the same parameters and the same data) will be close to the true intractable  $p(y_{1:T} | \theta)$ .

The use of the particle filter as an estimator of the likelihood has opened up possibilities for combining it with another branch of Monte Carlo methods, namely Markov chain Monte Carlo (MCMC). This combination allows for inferring not only unobserved states  $x_t$  but also unknown parameters  $\theta$  in nonlinear state-space models. One such successful idea is to construct a high-level procedure concerned with  $\theta$ , and then run the particle filter to estimate the likelihood for different parameter values. The high-level procedure can be a Metropolis-Hastings algorithm [27], [37, Section 5], essentially an informed random walk in  $\Theta$ . The Metropolis-Hastings algorithm is constructed such that after sufficiently long time, the random walk (the ‘chain’) in  $\Theta$  will have a trace that is a sample from the sought distribution  $p(\theta | y_{1:T})$ . The original Metropolis-Hastings algorithm assumes though that the target distribution can be evaluated exactly. In the state-space learning problem it means that the estimate  $z$  from the particle filter would not be sufficient for a valid Metropolis-Hastings algorithm. However, it has lately been shown [1] that valid algorithms can be constructed based also on stochastic estimates with certain properties, which provides the ground for the particle (marginal) Metropolis-Hastings (PMH, [2]) algorithm. We will not go into further details here, but refer to [37]. We do, however, note that the less variance in the estimates  $z$ , the better mixing properties of the Markov chain. Mixing is a measure of the ability of the algorithm to explore  $\Theta$ , and the worse mixing the bigger risk of the random walk to get stuck for long periods of time, wasting computational resources.

An alternative partner for the particle filter instead of the MCMC, is the sequential Monte Carlo (SMC). Interestingly enough, SMC is a family of methods that has been developed as a generalization of the particle filter. One SMC method is the SMC sampler [14], which alternatively can be employed to handle the unknown  $\theta$  instead of Metropolis-Hastings. The SMC sampler will then query the particle filter for likelihood estimates  $z$  for different values of  $\theta$ . The SMC sampler itself is similar to a particle filter, propagating its  $N_\theta$  samples  $\{\theta^j\}_{j=1}^{N_\theta}$  through a sequence ending up in the posterior  $p(\theta | y)$ . The sequence through which the samples of  $\theta$  is propagated can be a so-called tempering sequence. With a certain choice of tempering sequence, the nested construction of particle filter and SMC sampler has been termed SMC<sup>2</sup> [9]. The method that we propose in this paper bears close resemblance to SMC<sup>2</sup>, but makes use of a different tempering sequence.

**2.2. Challenges with highly informative observations.** The particle filter is often used to provide estimates ( $z$ ) of the likelihood ( $p(y_{1:T} | \theta)$ ) in probabilistic learning methods. However, when there is (almost) no measurement noise and thus the highly informative observations, these estimates become poor due to the importance sampling mechanism inherent in the particle filter. In the bootstrap particle filter,  $N_x$  particles  $\{x_t^n\}_{n=1}^{N_x}$  are drawn from (1a), and then weighted by evaluating (1b). As long as there is at least one particle  $x_t^n$  which gives a reasonably high probability for the measurement  $y_t$  (and consequently gets assigned a large weight), the particle filter will provide a reasonable result, and the more such high-weight particles, the better (in terms of variance of the estimate  $z$ ). However, if no samples are drawn under which  $y_t$  could have been observed with reasonably high probability, the estimate  $z$  will be very poor. If there is very little measurement noise but a reasonable amount of process noise in the model, the chances of drawing any useful particles  $x_t^n$  are typically rather small. The problem may become even more articulated if the bootstrap particle filter is run with a parameter  $\theta$  which does not explain the measurements  $y_{1:T}$  well. The bottom line is that a model with highly informative observations causes the bootstrap particle filter to provide estimates  $z$  with high variance. This is in particular true for values of  $\theta$  that do not explain the measurements well. Considering that high variance of  $z$  implies bad performance in probabilistic learning algorithms for  $\theta$ , the model (1) is problematic to learn.

To this end, research has been done on how to improve the particle filter by drawing particles  $\{x_t^n\}_{n=1}^{N_x}$  not from (1a) but instead from a tailored proposal which also can depend on  $y_t$ , in order to adapt to the measurement  $y_t$  better and make more ‘well-informed’ particle draws. In order to retain consistency, the weight update is modified accordingly. Such adaption is not always simple, but proposed methods include the fully adapted auxiliary particle filter [31] (only possible for a limited set of model structures), the alive particle filter [16] and the bridging particle filter [13] (both computationally more costly). In this work, we will however not focus on this aspect, but rather on how inference about  $\theta$  can be constructed in order to as far as possible avoid running the particle filter for models with highly informative observations. Ultimately, our suggested approach could be combined with methods like the fully adapted, alive or bridging particle filter to push the limits even further.

**2.3. Tempering.** To construct inference algorithms, the computational trick of tempering (or annealing, as it also is called) has proven useful. The name tempering was originally used for a certain heat treatment method within metallurgy, but the term is also used in a figurative sense for a set of computational methods. The idea is to construct a ‘smooth’ sequence  $\{\pi_p(\theta)\}_{p=0}^P$  starting in a user-coosen initial function  $\pi_0(\theta)$  and ending in the target function  $\pi_P(\theta)$ . In our case, these functions are probability densities, and our target is  $\pi_P(\theta) = p(\theta | y_{1:T})$ , as illustrated in Figure 1a. There are several ways in which such a sequence can be constructed. We do not have a formal definition of ‘smooth’, but understand it as a sequence where every adjacent pair  $\{\pi_p(\theta), \pi_{p+1}(\theta)\}$  are point-wise similar. A tempering sequence  $\{\pi_p(\theta)\}_{p=0}^P$  can be used for probabilistic learning problems in the search for the maximum mode (or in optimization, the maximum) of the target  $\pi_P(\theta)$ . By tracking the evolution from the typically simple and unimodal  $\pi_0(\theta)$  to the potentially intricate and multimodal target  $\pi_P(\theta)$ , the risk of getting stuck in zero-gradient regions or in local optima is reduced, compared to standard methods starting directly in  $\pi_P(\theta) = p(\theta | y_{1:T})$ .

For state-space models, there are several generic choices for constructing tempering sequences ending up in a posterior  $p(\theta | y_{1:T})$ . One choice (with  $P = T$ ) is the data-tempered sequence  $\pi_p(\theta) = p(\theta | y_{1:p})$ , which gives a sequence starting in the prior  $p(\theta)$  and, by sequentially including one additional measurement  $y_p$ , eventually ending in  $p(\theta | y_{1:T})$ . Typically, the landscape of  $p(\theta | y_{1:t})$  does not change dramatically when including one extra measurement, which ensures the smoothness. Another choice is found by first noting that  $p(\theta | y_{1:T}) \propto p(y_{1:T} | \theta)p(\theta)$ , and then making the choice  $\pi_p(\theta) = p(y_{1:T} | \theta)^{p/P}p(\theta)$ . Such a sequence also starts in the prior  $p(\theta)$  and ends, with  $p = P$ , in the posterior  $p(\theta | y_{1:T})$ . We will in this paper introduce a new sequence that is tailored for state-space models with highly informative observations.

**2.4. Using a tempering sequence in an SMC sampler.** A tempering sequence  $\{\pi_p(\theta)\}_{p=0}^P$  can be used in an SMC sampler to produce samples from  $\pi_P(\theta) = p(\theta | y_{1:T})$ . The idea underlying the SMC sampler is to propagate a set of  $N_\theta$  samples  $\{\theta^j\}_{j=1}^{N_\theta}$  along the tempering sequence, and—thanks to the smoothness of the sequence—gain a high computational efficiency by generating samples primarily in the most relevant part of  $\Theta$ , compared to more basic sampling schemes such as importance sampling. One version of the SMC sampler is a sequential iteration of importance sampling and resampling on the sequence  $\{\pi_p(\theta)\}_{p=0}^P$ , proceeding as follows: samples  $\{\theta^j\}_{j=1}^{N_\theta}$  are initially drawn from  $\pi_0(\theta)$ , and assigned importance weights  $W_1^j$  from the ratio  $\frac{\pi_1(\theta^j)}{\pi_0(\theta^j)}$ . The samples are then resampled and moved around in the landscape of  $\pi_1(\theta)$  with one or a few step with Metropolis-Hastings. After  $P$  such iterations, samples from  $\pi_P(\theta)$  are obtained. An illustration can be found in Figure 1b.

A reader familiar with PMH may understand this use of the SMC sampler as a manager of  $N_\theta$  parallel PMH chains, which aborts and duplicates the chains in order to optimize the overall performance<sup>1</sup>.

### 3. SOLUTION STRATEGY

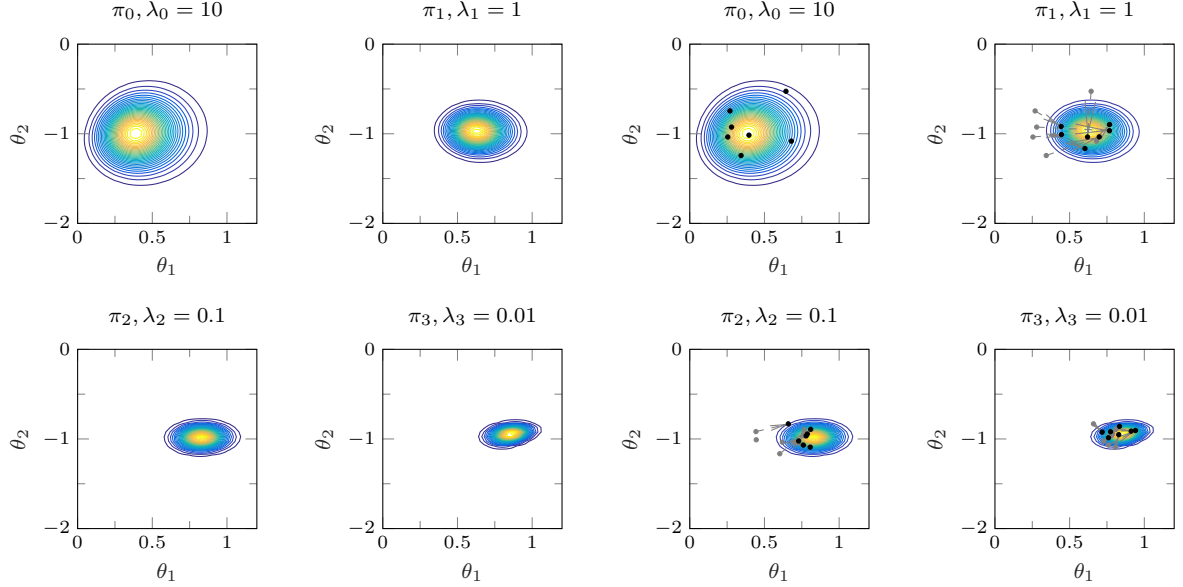
Provided the background on particle filters, tempering and SMC samplers, we are now ready to put our proposed solution together. In the following section, we will first propose a novel tempering idea suited for learning parameters  $\theta$  in models on the form (1), and then explore how the tempering pace can be automatically adapted to the problem. We will thereafter summarize an overview of the proposed algorithm, and detail some connections to existing literature.

**3.1. A tempering sequence for our problem.** Our aim is to infer the posterior  $p(\theta | y_{1:T})$  for the model (1). The absence of measurement noise in (1b) gives the likelihood estimate  $z$  from the bootstrap particle filter a high variance (in particular for values of  $\theta$  not explaining the data well), which is a problem when seeking  $p(\theta | y_{1:T})$ . We therefore suggest to introduce the modified model

$$(3a) \quad p(x_t | x_{1:t-1}, \theta) = f(x_t | x_{t-1}, u_{t-1}, \theta),$$

$$(3b) \quad y_t = g(x_t) + e_t, \quad e_t \sim \mathcal{N}(0, \lambda_p).$$

<sup>1</sup>A subtle but important difference to vanilla PMH is that a PMH chain is typically initialized arbitrarily, run until it converges, and then is its transient behavior (the burn-in period) discarded. In the SMC sampler, however, all chains are ‘warm-started’ thanks to the resampling mechanism, and it is therefore *not* relying on asymptotics in  $K \rightarrow \infty$  for converging.



(A) A tempering sequence shown by level curves for  $\pi_p(\theta)$ . The tempering sequence used in this figure is the sequence (4) we will propose in a SMC sampler which propagates samples (black dots) through the target distributions. (B) The problem of localizing the peak of  $\pi_3(\theta)$  can be solved with the sequence (4) we will propose in a SMC sampler which propagates samples (black dots) through the target distributions. For a linear state-space model with two unknown parameters  $\theta_1$  and  $\theta_2$ . With decreasing  $\lambda_p$ , which is the variance of an ‘imagined’ Gaussian measurement noise, tempering is obtained, starting in a distribution with a broad support, and ending in a more narrow and peaky distribution. The problem of large uninteresting regions becomes particularly articulated in high dimensional problems.

FIGURE 1. An illustration of the tempering idea. The underlying model for this problem will later be used as an example in Section 5.1.

This model has an ‘imagined’ Gaussian<sup>2</sup> measurement noise with variance  $\lambda_p$ , and our original model (1) is recovered for  $\lambda_p = 0$ . We denote the posterior distribution under this model as  $p(\theta | y_{1:T}, \lambda_p)$ , and the corresponding likelihood  $p(y_{1:T} | \theta, \lambda_p)$ . Furthermore, we will define a decreasing sequence of  $\lambda_p$ , such that  $\lambda_P = 0$ , and get a tempering sequence

$$(4) \quad \pi_p(\theta) = p(\theta | y_{1:T}, \lambda_p) \propto p(y_{1:T} | \theta, \lambda_p)p(\theta),$$

which we have illustrated in Figure 1. In this sequence, the target distribution (at  $p = P$ ) indeed becomes  $\pi_P(\theta) = p(\theta | y_{1:T}, \lambda_P = 0) = p(\theta | y_{1:T})$ , i.e., the posterior for  $\theta$  in the original model (1), the problem we study in this paper. Such a tempering sequence bears clear resemblance to the ABC methodology proposed in [15]. However, to the best of the authors knowledge, such a tempering sequence has not previously been studied in the context of state-space models.

We will use the tempering sequence 4 in an SMC sampler. For using an SMC sampler, we need to be able to evaluate  $\pi_p(\theta)$  up to proportionality. For this purpose, we propose to use the particle filter to estimate the likelihood  $p(y_{1:T} | \theta)$  (and assume  $p(\theta)$  can be evaluated). The algorithm will thus have a nested construction of SMC algorithms: the particle filter is used to generate likelihood estimates  $z_p$  for different values of  $\theta$  and  $\lambda_p$ , and the SMC sampler is used to infer  $\theta$  by keeping track of the samples  $\{\theta^j\}_{j=1}^{N_\theta}$  and deciding for which values of  $\theta$  to run the particle filter. However, to ease the presentation, we will throughout the rest of this section assume that we do have access to the likelihood  $p(y_{1:T} | \theta, \lambda_p)$  exactly. That is indeed the case if, for example, (1) is a linear Gaussian state-space model or a finite discrete hidden Markov model, in which cases the Kalman filter [35] and the forward-backward algorithm [7] would provide  $p(y_{1:T} | \theta, \lambda_p)$  exactly. We will, however, return (in Section 4) to the situation that we only have access

<sup>2</sup>The choice of Gaussian noise is only for convenience and clarity. Other choices, for example heavy-tailed distributions, are also possible. The only requirement is that its density can be evaluated point-wise.

to stochastic estimates  $z_p$ , and expand the algorithm with a few more details to ensure theoretical soundness also for that general (and practically interesting) case.

**3.2. Automatically determining the tempering pace.** Choosing a good sequence  $\{\lambda_p\}_{p=1}^P$  is fundamental to the performance of the proposed algorithm. A sequence  $\{\lambda_p\}_{p=0}^P$  that is decreasing too fast will lead to rapid changes in the landscape of  $\pi_p(\theta) = p(\theta | y_{1:T}, \lambda_p)$ , which obstructs the SMC sampler and adds to the variance of the final estimate. On the other hand, a sequence  $\{\lambda_p\}_{p=0}^P$  that is decreasing too slowly will be a waste of computational power. To this end, we suggest to take inspiration from Del Moral et al. [15], where they tackle a somewhat similar problem with the same version of the SMC sampler. They argue that a good tempering sequence would yield an effective sample size (ESS, [25]) somewhat constant throughout the sequence  $p = 0, \dots, P$ . The ESS is defined as

$$(5) \quad \text{ESS} \left( \{W_p^j\}_{j=1}^{N_\theta} \right) = \left( \sum_{j=1}^{N_\theta} \left( \frac{W_p^j}{\sum_{k=1}^{N_\theta} W_p^k} \right)^2 \right)^{-1},$$

where  $W_p^j$  denotes the importance weight of sample  $j$  from  $\pi_p(\theta)$ . The ESS takes values between 1 and  $N_\theta$ , with the interpretation that inference based on the  $N_\theta$  weighted samples is approximately equivalent to inference based on  $\text{ESS} \left( \{W_p^j\}_{j=1}^{N_\theta} \right)$  equally weighted samples. Consequently, if the weight of a single sample dominates all the other, the ESS is 1, and if all samples have equal weights, the ESS is  $N_\theta$ . Furthermore, Del Moral et al. [15] note that on their problem it is possible to solve the equation of setting  $\lambda_p$  (note that  $W_p^j$  depends on  $\lambda_p$ ) such that

$$(6) \quad \text{ESS} \left( \{W_p^j\}_{j=1}^{N_\theta} \right) = \alpha N_\theta,$$

where  $\alpha$  is some user-chosen coefficient between 0 and 1. (A similar adaption can also be found in [23].) It turns out, perhaps a bit surprisingly, that it is in fact possible to solve (6) also in our case when  $W_p^j$  depends on  $z_p^j$  from the particle filter, which in turn depends on  $\lambda_p$ . We postpone the details to the subsequent section where we discuss the details of the inner particle filter algorithm which defines the estimate  $z_p^j$ . By solving (6), we obtain an automated way to determine the tempering pace ‘on the fly’, i.e., automatically determining the value of each  $\lambda_p$  with the aim to achieve a constant ‘quality’ (constant ESS) of the Monte Carlo approximation in runtime.

The variance of the estimates  $z_p$  is likely to increase as  $p$  increases and  $\lambda_p$  approaches 0 (Section 2.2). The implications of an increased variance of  $z_p$  will be that fewer of the proposed samples will be accepted in Metropolis-Hastings, and the overall performance of the SMC sampler will deteriorate. It may therefore be necessary to terminate the sampler prematurely (at, say,  $\lambda_p = 0.01$  instead of the desired  $\lambda_p = 0$ ), and take the obtained samples as an approximate solution. One heuristic suggested by [15] for determining a suitable termination point is to monitor the rejection rate in the Metropolis-Hastings steps, and trigger a termination when it reaches a certain threshold.

**3.3. Proposed algorithm – preliminary version.** In Algorithm 1 we outline our proposed algorithm. We have in Algorithm 1 assumed that  $p(y_{1:T} | \theta, \lambda_p)$  can be evaluated exactly. However, in the general case of a nonlinear state-space model the particle filter has to be used, which results in an unbiased stochastic estimate  $z_p \approx p(y_{1:T} | \theta, \lambda_p)$ . We will address this fully in Section 4.

As mentioned earlier, a parallel to our problem with no measurement noise can be found in the literature under the heading approximate Bayesian computations (ABC, [3]). In ABC, the idea is to simulate data  $\hat{y}(\theta)$  from a model and compare it to the recorded data  $y$ . ABC, however, is originally not formulated for state-space models, even though recent such contributions have been made [22, 11]. The introduction of an imagined measurement noise in our problem can be seen as an ABC-type of idea, but since the imagined measurement noise interacts with the particle filter (our analogy to simulate new data  $\hat{y}(\theta)$ ), our method does not qualify as a standard ABC solution.

Another closely related algorithm is the SMC<sup>2</sup> algorithm [9]. SMC<sup>2</sup> is also an SMC sampler using the particle filter to estimate the likelihood  $z$ , but it makes use of a data-tempered sequence (Section 2.3) instead of tempering based on imagined measurement noise (4). For the problem of learning the parameters  $\theta$  in (1), the particle filter is likely to face troubles for small values of measurement noise  $\lambda_p$ . For our proposed algorithm, this can be handled by terminating the algorithm prematurely if necessary. Such a resort is not possible with a data-tempered sequence as in SMC<sup>2</sup>, since the problems with poor estimates  $z$  from the particle filter would be faced already from the first step of a data-tempered sequence.

**Output:** Samples  $\{\theta^j\}_{j=1}^{N_\theta}$  from  $p(\theta | y_{1:T}, \lambda_p)$ .

Set  $p \leftarrow 0$  and  $\lambda_0$  large.

Sample initial  $\{\theta^j\}_{j=1}^{N_\theta} \sim p(\theta | y_{1:T}, \lambda_0)$  using, e.g., Metropolis-Hastings.

**while**  $\lambda$  not sufficiently small **do**

    Update  $p \leftarrow p + 1$ .

    Let  $\omega^j \leftarrow p(y_{1:T} | \theta^j, \lambda_{p-1})p(\theta^j)$ .

    Find  $\lambda_p$  such that  $\text{ESS}(\{\omega^j\}_{j=1}^{N_\theta}, \{\tilde{\omega}^j = p(y_{1:T} | \theta^j, \lambda_p)p(\theta^j)\}_{j=1}^{N_\theta}) = \alpha \cdot N_\theta$ .

    Let  $\tilde{\omega}^j \leftarrow p(\theta^j | y_{1:T}, \lambda_p)$ .

    Draw  $a^j$  with  $\mathbb{P}(a^j = k) \propto \frac{\tilde{\omega}^k}{\omega^k}$ .

    Sample  $\theta^j \leftarrow \text{Metropolis-Hastings}(\lambda_p, \theta^j)$ .

**end**

**Function**  $\text{ESS}(\{\omega^j\}_{j=1}^{N_\theta}, \{\tilde{\omega}^j\}_{j=1}^{N_\theta})$

    Let  $W^j \leftarrow \frac{\tilde{\omega}^j}{\omega^j}$ .

**return**  $\left( \sum_{j=1}^{N_\theta} \left( W^j / \sum_{k=1}^{N_\theta} W^k \right)^2 \right)^{-1}$

**Function**  $\text{Metropolis-Hastings}(\lambda_p, \theta^j)$

    Propose a new  $\theta' \sim q(\cdot | \theta^j)$ .

    Sample  $d \leftarrow \mathcal{U}_{[0,1]}$ , i.e., uniformly on the interval  $[0, 1]$ .

**if**  $d < \frac{p(y_{1:T} | \theta', \lambda_p)p(\theta')}{p(y_{1:T} | \theta^j, \lambda_p)p(\theta^j)} \frac{q(\theta^j | \theta')}{q(\theta' | \theta^j)}$  **then**

        Accept  $\theta^j \leftarrow \theta'$ .

**end**

**return**  $\theta^j$

(Lines with  $j$  are for all  $j = 1, \dots, N_\theta$ ).

**Algorithm 1:** Strategy for particle-filter based learning of  $\theta$  in (1)

#### 4. FULL ALGORITHM AND DETAILS

In this section, we will first consider how to initialize the algorithm, and thereafter the details concerning the particle filter required for the adaption of  $\lambda$ . Next, we present the proposed algorithm in full details and fully address the fact that the particle filter only provides stochastic estimates  $z$ , whereas Algorithm 1 requires that  $p(y_{1:T} | \theta)$  can be evaluated exactly. The key is to consider the proposed algorithm to be sampling from an extended space explicitly encoding the randomness in the estimator  $z$ , and thereby reduce the problem to a standard SMC algorithm.

**4.1. Initialization.** To initialize the SMC sampler properly, samples  $\{\theta^j\}_{j=1}^{N_\theta}$  from  $p(\theta | y_{1:T}, \lambda_0)$  are required. However, that distribution is typically not available to draw samples from. To this end, PMH [37] (or SMC<sup>2</sup>) can be used. Since  $\lambda_0$  is user-chosen, we can choose it big enough such that we can obtain low-variance estimates  $z_0$  and make sure  $p(y_{1:T} | \theta, \lambda_0)$  has a broad support. For such values of  $\lambda_0$  PMH is feasible, even though the sought  $p(\theta | y_{1:T})$  is problematic. However, the use of Metropolis-Hastings inside the SMC sampler makes the algorithm in practice somewhat ‘forgiving’ with respect to initialization, and it may for practical purposes suffice to initialize the algorithm with samples  $\{\theta^j\}_{j=1}^{N_\theta}$  that are only approximate samples from  $p(\theta | y_{1:T}, \lambda_0)$  obtained using, e.g., some suboptimal optimization-based method.

**4.2. Re-visiting the particle filter.** We have so far not fully justified the use of the particle filter inside the proposed algorithm. The particle filter provides only a *stochastic* estimate  $z_p$  of  $p(y_{1:T} | \theta, \lambda_p)$ , and the  $\lambda_p$ -adaption requires that we can solve (6),  $\text{ESS}(\{W_p^j\}_{j=1}^{N_\theta}) = \alpha N_\theta$ , where  $W_p^j$  depends on the ratio between  $z_p$  and  $z_{p-1}$ , in turn depending on  $\lambda_p$  and  $\lambda_{p-1}$ , respectively. Both estimates,  $z_p$  and  $z_{p-1}$ , are stochastic, which seems not to allow for a well-defined numerical solution to the equation (6). This also implies that the weights  $W^j$  in the SMC sampler are random themselves. The latter problem of stochastic weights in a SMC is, however, already studied in the literature [18], whereas solving (6) is novel in this work.

**Input:** State space model  $f(\cdot | \cdot, \theta)$ ,  $g(\cdot)$ ,  $\lambda_p$ ,  $p(x_1)$ , and data  $y_{1:T}$ .

**Output:**  $x_{1:T}, a_{2:T}$

Sample  $x_1^n \sim p(x_0)$ .

Compute  $w_1^n \leftarrow \mathcal{N}(y_1 | g(x_1^n), \lambda_p)$ .

**for**  $t = 2$  **to**  $T$  **do**

    Sample  $a_t^n$  with  $\mathbb{P}(a_t^n = j) \propto w_{t-1}^j$ .

    Sample  $x_t^n \sim f(x_t | x_{t-1}^{a_t^n}, \theta)$ .

    Compute  $w_t^n \leftarrow \mathcal{N}(y_t | g(x_t^n), \lambda_p)$ .

**end**

All operations are for  $n = 1, \dots, N_x$ .

**Algorithm 2:** Bootstrap particle filter

The key point for solving (6) in our context with particle filters, and also to theoretically justify the random weights, is to consider the outcome of the particle filter (Algorithm 2) to be all its internal random variables,  $\{\mathbf{x}_{1:T}, \mathbf{a}_{2:T}\} \triangleq \{x_{1:T}^n, a_{2:T}^n\}_{n=1}^{N_x}$ , rather than only  $z$ . By doing so, we can explicitly handle all randomness in the particle filter, and understand our proposed algorithm as a standard algorithm on the extended space  $\Theta \times \mathbf{X}^{N_x T} \times \mathbf{A}^{N_x(T-1)}$ , where  $\mathbf{X}$  in which  $x_t$  lives, and similar for  $\mathbf{A}$  and  $a_t$ . We will come back to this formalism, but let us first give a more intuitive view on the construction.

In solving (6), we would like to run the particle filter once (using  $\lambda_{p-1}$ ), and then afterwards decide on a  $\lambda_p$  such that (6) is fulfilled. The random variables in the particle filter,  $\{\mathbf{x}_{1:T}, \mathbf{a}_{2:T}\}$ , are however random, *but with a certain distribution* determined by Algorithm 2 and  $\lambda_p$ . That is, if we are given samples  $\{\mathbf{x}_{1:T}, \mathbf{a}_{2:T}\}$ , we could compute the probability (density) of that very particle filter to be executed. In particular, by inspection of Algorithm 2, we realize that *if* the ancestor variables  $\mathbf{a}_{2:T}$  were fixed,  $\lambda_p$  would not affect  $\mathbf{x}_{1:T}$ , but only the computation of  $z$ . Thus, if we run a particle filter with a measurement noise model with variance  $\lambda_{p-1}$  and save  $\{\mathbf{x}_{1:T}, \mathbf{a}_{2:T}\}$ , we may afterwards compute the probability (density) of the resampling to have happened had it been run with a measurement noise model with variance  $\lambda_p$  instead<sup>3</sup>. This turns out to be enough for evaluating  $\text{ESS}(\{W_p^j\}_{j=1}^{N_\theta})$  conditionally on  $\{x_{1:T}, a_{2:T}\}_{n=1}^{N_x}$ , which can be used to solve (6) using a numerical search, such as a bisection method.

We summarize our proposed method by the complete Algorithm 3. Continuing the extended space motivation from Section 4.2, Algorithm 3 can in its most compact form be seen as a standard SMC sampler on the extended space  $\Theta \times \mathbf{X}^{N_x T} \times \mathbf{A}^{N_x(T-1)}$  with target distribution at iteration  $p$

$$(7) \quad p(\theta, \mathbf{x}_{1:T}, \mathbf{a}_{2:T} | y_{1:T}, \lambda_p) \propto$$

$$\left( \prod_{t=1}^T \sum_{n=1}^{N_x} g(y_t | x_t^n, \lambda_p) \right) \left( \prod_{t=1}^{T-1} \prod_{n=1}^{N_x} \underbrace{\mathbb{P}(a_{t+1}^n | \{w_t^n\}_{n=1}^{N_x})}_{= \frac{g(y_t | x_t^{a_{t+1}^n}, \lambda_p)}{\sum_{n=1}^{N_x} g(y_t | x_t^n, \lambda_p)}} \right) \left( \prod_{t=1}^{T-1} \prod_{n=1}^{N_x} f(x_{t+1}^n | x_t^{a_{t+1}^n}, \theta) \right).$$

From this, Algorithm 3, and in particular the particle filter (Algorithm 2 as well as the weighting function  $w$  in Algorithm 3, can be derived.

The previous paragraph can be understood as follows: the particle filter algorithm itself contains random elements. If we consider all randomness in the particle filter explicitly as random variables, i.e., consider  $\mathbf{x}_{1:T}, \mathbf{a}_{2:T}$  and not just  $z$ , Algorithm 3 is a standard SMC sampler [14] for the distribution (7). This implies that available theoretical guarantees and convergence results (e.g., [14, 8, 12]) applies also to our construction when the  $\lambda_p$  sequence is fixed. When  $\lambda_p$  is selected adaptively these results do not readily apply. [5] have established convergence results for adaptive SMC algorithms in a related setting, and these results could possibly be extended to the adaptive scheme proposed in this article. Note also that no practical problems caused by the proposed adaptation have been encountered in the numerical examples.

<sup>3</sup>For this answer not to be exactly 0 forbiddingly often, multinomial resampling has to be used.



**Output:** Samples  $\{\theta^j\}_{j=1}^{N_\theta}$  from  $p(\theta | y_{1:T}, \lambda)$ .

Set  $p \leftarrow 0$  and  $\lambda_0$  large.

Sample initial  $\{\theta^j\}_{j=1}^{N_\theta} \sim p(\theta | y_{1:T}, \lambda_0)$  using, e.g., particle Metropolis-Hastings.

Run a particle filter for each  $\theta^j$ , and save  $\xi^j \triangleq \{x_{1:T}^n\}_{n=1}^{N_x}, \{a_{2:T}^n\}_{n=1}^{N_x}$ .

**while**  $\lambda_p$  not sufficiently small **do**

Update  $p \leftarrow p + 1$ .

Let  $\omega^j \leftarrow w(\lambda_{p-1}, \theta^j, \xi^j)$ .

Find  $\lambda_p$  such that  $\text{ESS}(\{\omega^j\}_{j=1}^{N_\theta}, \{w(\lambda_p, \theta^j, \xi^j)\}_{j=1}^{N_\theta}) = \alpha \cdot N_\theta$ .

Let  $\tilde{\omega}^j \leftarrow w(\lambda_p, \theta^j, \xi^j)$ .

Resample the  $(\theta, \xi)$ -particles using weights  $\propto \frac{\tilde{\omega}^k}{\omega^k}$ .

Sample  $\{\theta^j, \xi^j\} \leftarrow \text{Particle Metropolis-Hastings}(\lambda_p, \theta^j, \xi^j)$ .

**end**

**Function**  $w(\lambda_p, \theta^j, \xi^j)$

Let  $w_{t-1}^n \leftarrow g(y_t | x_t^n, \lambda_p)$

**return**  $p(\theta^j) \left( \prod_{t=1}^T \sum_{n=1}^{N_x} w_{t-1}^n \right) \left( \prod_{t=1}^{T-1} \prod_{n=1}^{N_x} \mathbb{P}(a_{t+1}^n | \{w_t^n\}_{n=1}^{N_x}) \right)$

**Function**  $\text{ESS}(\{\omega^j\}_{j=1}^{N_\theta}, \{\tilde{\omega}^j\}_{j=1}^{N_\theta})$

Let  $W^j \leftarrow \frac{\tilde{\omega}^j}{\omega^j}$  for every  $j$

**return**  $\left( \sum_{j=1}^{N_\theta} \left( W^j / \sum_{k=1}^{N_\theta} W^k \right)^2 \right)^{-1}$

**Function**  $\text{Particle Metropolis-Hastings}(\lambda_p, \theta^j, \xi^j)$

Let  $z_p^j \leftarrow \prod_{t=1}^T \sum_{n=1}^{N_x} g(y_t | x_t^i, \lambda_p)$  (with  $x_t^i$  from  $\xi^j$ )

Propose a new  $\theta' \sim q(\cdot | \theta^j)$

Run a particle filter with  $\theta'$  and save  $\xi'$

Let  $z_p' \leftarrow \prod_{t=1}^T \sum_{n=1}^{N_x} g(y_t | x_t^i, \lambda_p)$  (with  $x_t^i$  from  $\xi'$ )

Sample  $d \leftarrow \mathcal{U}_{[0,1]}$ , i.e., uniformly on the interval  $[0, 1]$ .

**if**  $d < \frac{z_p' p(\theta')}{z_p^j p(\theta^j)} \frac{q(\theta^j | \theta')}{q(\theta' | \theta^j)}$  **then**

Update  $\theta^j \leftarrow \theta', \xi^j \leftarrow \xi'$

**end**

**return**  $\theta^j, \xi^j$

**Algorithm 3:** Particle-filter based learning of  $\theta$  in (1)

## 5. NUMERICAL EXPERIMENTS

In this section, we provide three numerical experiments evaluating the proposed method from various perspectives. First, we start with a simple numerical example with a linear state-space model subject to Gaussian noise (implicitly introduced by Figure 1) to illustrate the main ideas presented by Algorithm 1. We then consider a more challenging nonlinear example, where we compare our proposed method to the PMH algorithm [2] and SMC<sup>2</sup> [9]. Finally we consider the challenging Wiener-Hammerstein benchmark problem [39]. The code for the examples is available via the first author's homepage.

**5.1. Toy example.** We consider the linear state-space model on the form

$$(8a) \quad x_{t+1} = \begin{bmatrix} 1 & \theta_1 \\ 0 & 0.1 \end{bmatrix} x_t + \begin{bmatrix} \theta_2 \\ 0 \end{bmatrix} u_t + v_t, \quad v_t \sim \mathcal{N}(0, I_2)$$

$$(8b) \quad y_t = \begin{bmatrix} 1 & 0 \end{bmatrix} x_t,$$

where  $\theta \triangleq \{\theta_1, \theta_2\}$  are the unknown parameters and  $I_2$  denotes the identity matrix of dimension 2. This model was used to produce Figure 1, where the propagation of samples  $\{\theta^j\}_{j=1}^{N_\theta}$  was illustrated. Since this model is linear and

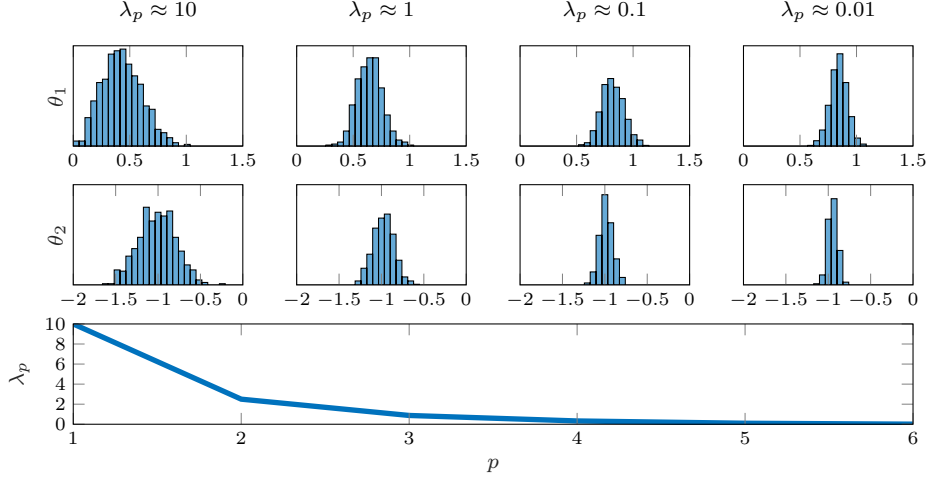


FIGURE 2. Result from applying Algorithm 1 to  $T = 200$  data points from the model (8). The upper panels shows how the marginals of the samples contracts as  $\lambda_p \rightarrow 0$  (cf. Figure 1), and the lower panel shows the sequence  $\{\lambda_p\}_{p=1}^P$  automatically determined by our algorithm in an adaptive manner.

Gaussian, the computation of the likelihood  $p(y_{1:T} | \theta, \lambda_p)$  can be done exact<sup>4</sup> and no particle filter (with its potential problem with small measurement noise variance) is needed. Thus, Algorithm 1 can be applied directly, by using the Kalman filter to exactly compute  $p(y_{1:T} | \theta, \lambda_p)$ . We now demonstrate Algorithm 1 by applying it to  $T = 200$  data points simulated from (8). The imagined measurement noise  $\lambda_p$  is automatically adapted such that ESS  $\approx 0.5$  at each step. The priors for both parameters are taken as uniform on  $[0, 2.5] \times [0, 2.5]$ . The resulting (marginal) posteriors are summarized in Figure 2 and shows that the automatic tempering seems to work as expected. It is also instructive to compare Figure 2 to Figure 1.

The main motivation behind our work was indeed to overcome the computational difficulties for the particle filter when the variance of the measurement noise is very small. However, for probabilistic learning of  $\theta$  also in linear Gaussian models where the exact Kalman filter can be applied, sampling methods are still useful for learning  $\theta$ , see, e.g. [28, 40] for the use of Metropolis-Hastings and Gibbs samplers, respectively. Our proposed algorithm thus presents another alternative also for linear systems, based on the SMC sampler.

**5.2. A more challenging example and comparison.** We now consider the following state-space model

$$(9a) \quad x_{t+1} = \text{atan}(x_t) + \theta_1 u_t + v_t, \quad v_t \sim \mathcal{N}(0, 1),$$

$$(9b) \quad y_t = |x_t| + \theta_1 \theta_2 + e_t, \quad e_t \sim \mathcal{N}(0, 10^{-2}).$$

This model, with a 1-dimensional state space, has as an exogenous input  $u_t$ , a significant amount of process noise  $v_t$  and an almost negligible measurement noise  $e_t$ . From this model,  $T = 300$  data points were simulated and the two unknown parameters  $\theta = \{\theta_1, \theta_2\}$  are to be learned from the data  $\{y_{1:T}, u_{1:T}\}$  with uniform priors. The input  $u_{1:T}$  is taken as a realization of a white noise random process.

The relatively short data record together with the presence of  $\theta_2$  only in the product  $\theta_1 \theta_2$  in (9b) suggest there is a certain amount of uncertainty present in the problem, which we expect to be reflected in the posterior. However, the highly informative observations makes this a rather challenging problem for the standard methods.

We apply our proposed method, and compare it to PMH [14] and SMC<sup>2</sup> [9] on this problem. In all algorithms, we use the bootstrap particle filter (Algorithm 2) with  $N_x = 300$ , and a simple random walk proposal. Furthermore, we let  $N_\theta = 200$ ,  $K = 10$  and  $\alpha = 0.3$  in Algorithm 3, as well as its counterpart in SMC<sup>2</sup>, and we run PMH until 100 000 samples were obtained. For our proposed algorithm, we adopt a similar heuristic as [15] and terminate the tempering once the acceptance rate in the Metropolis-Hastings procedure goes below 2.5%.

<sup>4</sup>The choice of a linear and Gaussian model also made it possible to exactly plot the contours in Figure 1.

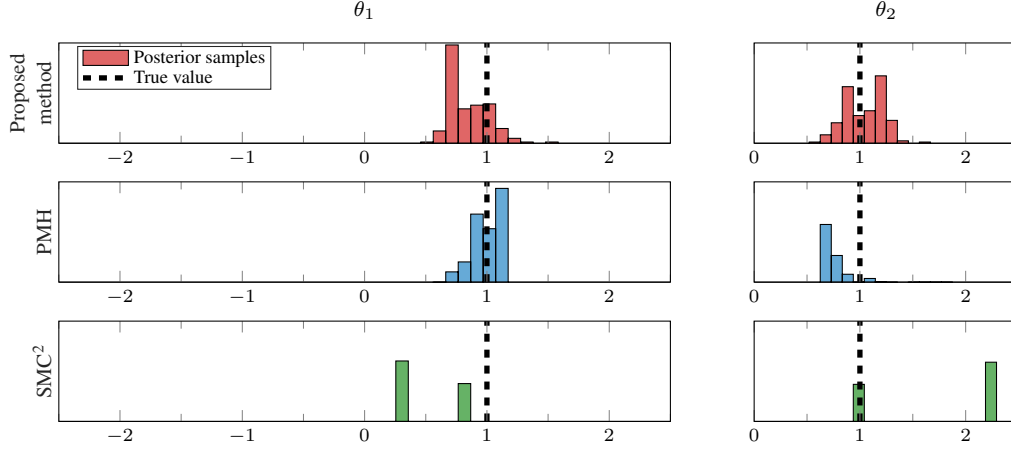


FIGURE 3. Posterior samples from the problem in Section 5.2. The probability of acceptance in the Metropolis-Hastings mechanism is very low due to the model (9) with highly informative observations, which gives an high variance in the estimates  $z$ . Neither PMH not  $\text{SMC}^2$  therefore explore the posterior well (compare, in particular,  $\theta_2$ ), whereas the proposed method shows a superior result thanks to the proposed tempering scheme, which adds an imagined measurement noise to the model which gives computational advantages with less variance in  $z_p$ .

The obtained posterior samples are shown in Figure 3. As clearly can be seen, the mixing of PMH is rather poor since it has not managed to explore the posterior as well as our proposed method. This also inherits into  $\text{SMC}^2$ , which performs even worse in this problem. Since the tempering in our proposed method, however, follows a sequence of decreasing imagined measurement noise, which terminates once the mixing becomes too bad, it does not suffer from the same problem. However, for this reason, the proposed method does not provide samples exactly from  $p(\theta | y_{1:T})$ , but  $p(\theta | y_{1:T}, \lambda = 0.17)$ .

The settings of PMH can indeed be optimized by using more clever proposals than random walks (see, e.g., [10] for an overview) and methods for reducing the variance of  $z$  (such as adapted or bridging particle filter [13, 31, 16]). Such adaption would indeed push the performance further. However, all three methods compared make use of the particle filter for estimating  $z$  in the Metropolis-Hastings ratio, and such tuning is therefore not crucial in a relative comparison between the methods.

**5.3. The Wiener-Hammerstein benchmark with process noise.** The Wiener-Hammerstein benchmark [39] is a recent benchmark problem for system identification which is a special case of the model (1). This problem has also served as the motivating problem for us to propose this method. The benchmark is implemented as an electronic circuit, and the benchmark problem is to use recorded data from the system to estimate a model which is able to imitate the behavior of the electric circuit well. The system can be described as a Wiener-Hammerstein system, i.e., a linear dynamical system, a static nonlinearity, and then another linear dynamical system in series. This is by now a fairly well-studied model, see e.g. [19, 38, 6, 4] for earlier work on this model. There was also a relatively recent special section devoted to an earlier Wiener-Hammerstein benchmark problem in Control Engineering Practice in 2012 [21]. The key challenge with the new benchmark is that there is a significant amount of process noise present.

The input to the system is a (known) signal entering into the first linear system. There is also an (unknown) colored process noise present, which enters directly into the nonlinearity. The measurements are of rather high quality, so there is very little measurement noise (when compared to the process noise) which makes the measurements highly informative. The system is summarized in Figure 4.

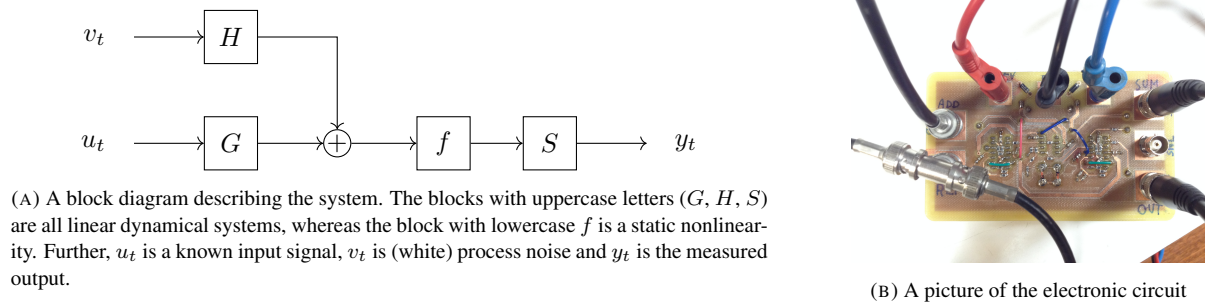


FIGURE 4. The Wiener-Hammerstein benchmark system.

	RMSE of proposed method	RMSE of initial model
Sweptsine	0.014	0.039
Multisine	0.015	0.038

TABLE 1. Wiener-Hammerstein benchmark: The root mean square error (RMSE) of the simulation error on the provided test data sets.

The structure of the Wiener-Hammerstein benchmark system can be brought into the state-space formalism as

$$\begin{aligned}
 \begin{bmatrix} x_{t+1}^G \\ x_{t+1}^H \\ x_{t+1}^S \end{bmatrix} &= \begin{bmatrix} A^G & 0 & 0 \\ 0 & A^H & 0 \\ 0 & 0 & A^S \end{bmatrix} \begin{bmatrix} x_t^G \\ x_t^H \\ x_t^S \end{bmatrix} + \begin{bmatrix} B^G \\ 0 \\ 0 \end{bmatrix} u_t + \begin{bmatrix} 0 \\ B^H \\ 0 \end{bmatrix} v_t \\
 (10a) \quad &+ \begin{bmatrix} 0 \\ 0 \\ B^S \end{bmatrix} \sum_{m=1}^M c^{(m)} \phi^{(m)}(C^G x_t^G + C^H x_t^H + D^G u_t), \\
 (10b) \quad &y_t = C_S x_t^S + D_S v_t^S,
 \end{aligned}$$

where all  $A$ s are  $3 \times 3$ -matrices,  $B$ s are  $3 \times 1$ -matrices,  $C$ s are  $1 \times 3$ -matrices,  $D$ s are scalars,  $\{\phi^{(m)}\}$  is a Fourier basis function expansion (truncated at  $M = 10$ ), and  $v_t$  is a zero-mean scalar-valued white Gaussian process noise with unknown variance. Adjusting for the overparametrization of the linear state-space model, the effective number of unknown parameters is 32. For learning the parameters, a data set with  $T = 8192$  samples and  $u_t$  a faded multisine input<sup>5</sup> was used. We applied Algorithm 3 with  $N_\theta = 50$  and  $K = 10$ . For initialization purposes, an approximate model was found essentially using the ideas by [29] (which is computationally lighter, but cannot fully handle the presence of process noise, on the contrary to Algorithm 3).

The obtained results are presented in Table 1 and Figure 5, where they are reported according to the benchmark instructions [39], i.e., the simulation error for two test data sets measured on the system with no process noise present and a sweptsine and a multisine as input, respectively. For reference, the performance of the model used to initialize Algorithm 3 is also included. The results reported were obtained within a few hours on a standard personal computer.

The essentially non-existing measurement noise makes PMH (and hence also SMC<sup>2</sup>) incompatible with this problem.

<sup>5</sup>Available as WH\_MultisineFadeOut at <http://homepages.vub.ac.be/~mschouke/benchmarkWienerHammerstein.html>.

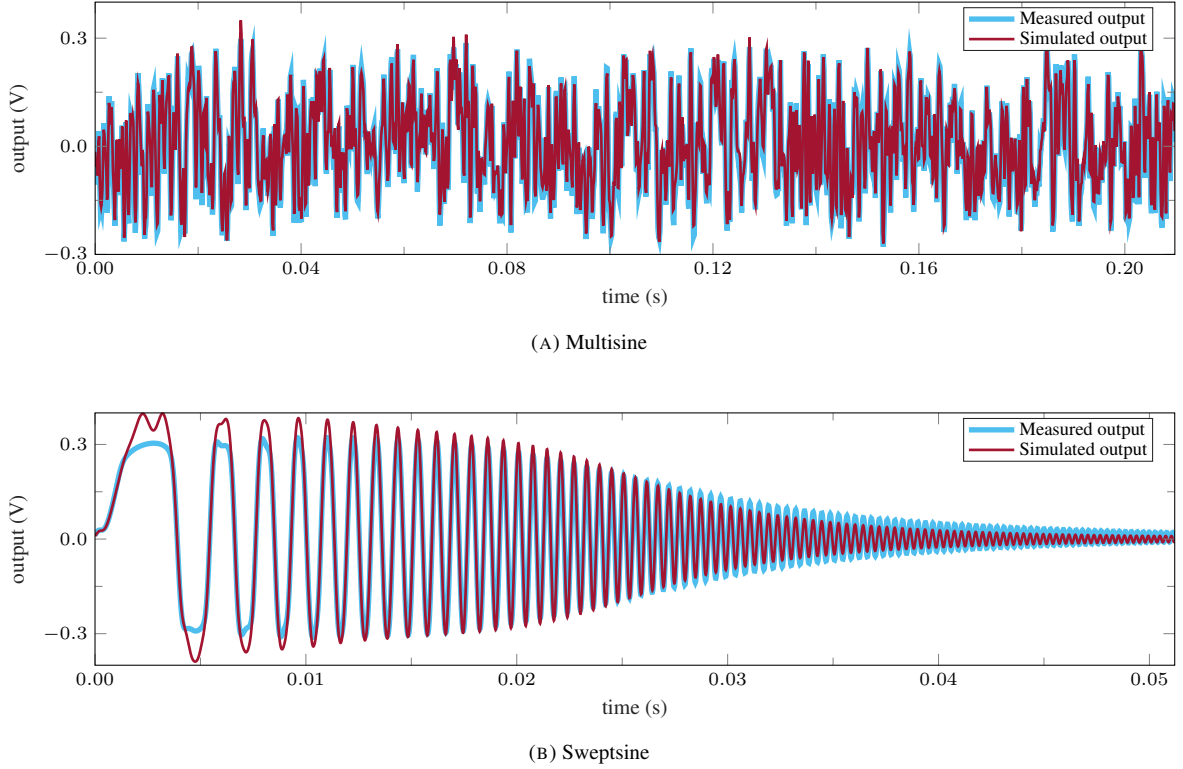


FIGURE 5. Simulated output from the model (red line) versus the recorded output (blue line) for the Wiener-Hammerstein benchmark. The test data sets are recorded with no process noise, as opposed to the training data sets that were used for learning the model.

## 6. DISCUSSION

We have proposed an algorithm for probabilistic learning of unknown parameters in models of the structure (1), i.e., state-space models with highly informative observations. Our proposed algorithm is either possible to understand as an ABC-inspired methodology, or as an alternative tempering in an SMC sampler (akin to SMC<sup>2</sup>). Its theoretical justification follows from understanding it as a standard SMC sampler on an extended space, and well established theoretical guarantees are thus available.

For optimal performance, our proposed method should be combined with methods for variance-reduction of the estimate  $z$ , such as the adapted or bridging particle filter [31, 13, 16]. The combination with such methods would indeed be interesting to explore further. However, in many models is the increased variance of  $z_p$  as  $\lambda_p \rightarrow 0$  unavoidable, and the use of such methods may indeed push limits, but does not change the fundamental problem.

For further research, connections with the idea of variational tempering [26] could possibly be of interest to explore. It is also not obvious that the ESS criterion (6) is the optimal criterion for deciding a well-performing tempering within the SMC sampler, and other alternatives could be studied and compared.

## ACKNOWLEDGEMENT

This research is financially supported by the Swedish Research Council via the projects *Probabilistic modeling of dynamical systems* (contract number: 621-2013-5524) and *Learning of Large-Scale Probabilistic Dynamical Models* (contract number: 2016-04278), and the Swedish Foundation for Strategic Research (SSF) via the project *ASSEMBLE*.

## REFERENCES

- [1] C. Andrieu and G. O. Roberts. The pseudo-marginal approach for efficient Monte Carlo computations. *Annals of Statistics*, 37(2):967–725, 2009.
- [2] C. Andrieu, A. Doucet, and R. Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010.
- [3] M. A. Beaumont, W. Zhang, and D. J. Balding. Approximate Bayesian computation in population genetics. *Genetics*, 162(4):2025–2035, 2002.
- [4] N. J. Bershad, P. Celka, and S. McLaughlin. Analysis of stochastic gradient identification of Wiener-Hammerstein systems for nonlinearities with Hermite polynomial expansions. *IEEE Transactions on Signal Processing*, 49(5):1060–1072, 2001.
- [5] A. Beskos, A. Jasra, N. Kantas, and A. Thiery. On the convergence of adaptive sequential Monte Carlo algorithms. *The Annals of Applied Probability*, 26(2):1111–1146, 2016.
- [6] S. A. Billings and S. Y. Fakhouri. Identification of systems containing linear dynamic and static nonlinear elements. *Automatica*, 18(1):15–26, January 1982.
- [7] O. Cappé, E. Moulines, and T. Rydén. *Inference in hidden Markov models*. Springer Series in Statistics. Springer, New York, NY, USA, 2005.
- [8] N. Chopin. Central limit theorem for sequential Monte Carlo methods and its application to Bayesian inference. *Annals of Statistics*, 36(6):2385–2411, 2004.
- [9] N. Chopin, P. E. Jacob, and O. Papaspiliopoulos. SMC<sup>2</sup>: an efficient algorithm for sequential analysis of state space models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 75(3):397–426, 2013.
- [10] J. Dahlin and T. B. Schön. Getting started with particle Metropolis-Hastings for inference in nonlinear models. *arXiv:1511:01707*, 2016.
- [11] T. A. Dean, S. S. Singh, A. Jasra, and G. W. Peter. Parameter estimation for hidden Markov models with intractable likelihoods. *Scandinavian Journal of Statistics*, 41(4):970–987, 2015.
- [12] P. Del Moral. *Feynman-Kac formulae: genealogical and interacting particle systems with applications*. Springer, New York, NY, US, 2004.
- [13] P. Del Moral and L. M. Murray. Sequential Monte Carlo with highly informative observations. *SIAM/ASA Journal on Uncertainty Quantification*, 3(1):969–997, 2015.
- [14] P. Del Moral, A. Doucet, and A. Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, 2006.
- [15] P. Del Moral, A. Doucet, and A. Jasra. An adaptive sequential Monte Carlo method for approximate Bayesian computation. *Statistics and Computing*, 22(5):1009–1020, 2012.
- [16] P. Del Moral, A. Jasra, A. Lee, C. Yau, and X. Zhang. The alive particle filter and its use in particle Markov chain Monte Carlo. *Stochastic Analysis and Applications*, 33(6):943–974, 2015.
- [17] A. Doucet and A. M. Johansen. A tutorial on particle filtering and smoothing: fifteen years later. In D. Crisan and B. Rozovsky, editors, *Nonlinear Filtering Handbook*, pages 656–704. Oxford University Press, Oxford, UK, 2011.
- [18] P. Fearnhead, O. Papaspiliopoulos, G. O. Roberts, and A. Stuart. Random-weight particle filtering of continuous time processes. *Journal of the Royal Statistical Society. Series B (Methodological)*, 72(4):497–512, 2010.
- [19] F. Giri and E. Bai, editors. *Block-oriented nonlinear system identification*, volume 404 of *Lecture notes in control and information sciences*. Springer, 2010.
- [20] N. J. Gordon, D. J. Salmond, and A. F. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE Proceedings F - Radar and Signal Processing*, pages 107–113, 1993.
- [21] H. Hjalmarsson, C. R. Rojas, and D. E. Rivera. System identification: A Wiener-Hammerstein benchmark. *Control Engineering Practice*, 20(11):1095–1096, 2012.
- [22] A. Jasra. Approximate Bayesian computation for a class of time series models. *International Statistical Review*, 83(3):405–435, 2015.
- [23] A. Jasra, D. A. Stephens, A. Doucet, and T. Tsagaris. Inference for Lévy-driven stochastic volatility models via adaptive sequential Monte Carlo. *Scandinavian Journal of Statistics*, 38(1):1–22, 2011.

- [24] N. Kantas, A. Doucet, S. S. Singh, J. M. Maciejowski, and N. Chopin. On particle methods for parameter estimation in state-space models. *Statistical Science*, 30(3):328–351, 2015.
- [25] A. Kong, J. S. Liu, and W. H. Wong. Sequential imputations and Bayesian missing data problems. *Journal of the American Statistical Association*, 89(425):278–288, 1994.
- [26] S. Mandt, J. McInerney, F. Abrol, R. Ranganath, and D. Blei. Variational tempering. In *Proceedings of the 19<sup>th</sup> International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 704–712, Cadiz, Spain, May 2016.
- [27] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1092, 1953.
- [28] B. Ninness and S. Henriksen. Bayesian system identification via Markov chain Monte Carlo techniques. *Automatica*, 46(1):40–51, 2010.
- [29] J. Paduart, L. Lauwers, J. Swevers, K. Smolders, J. Schoukens, and R. Pintelon. Identification of nonlinear systems using polynomial nonlinear state space models. *Automatica*, 46(4):647 – 656, 2010.
- [30] V. Peterka. Bayesian system identification. *Automatica*, 17(1):41–53, 1981.
- [31] M. K. Pitt and N. Shephard. Filtering via simulation: auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599, 1999.
- [32] M. K. Pitt, R. d. S. Silva, P. Giordani, and R. Kohn. On some properties of Markov chain Monte Carlo simulation methods based on the particle filter. *Journal of Econometrics*, 171(2):134–151, 2012.
- [33] C. P. Robert. *The Bayesian choice: from decision-theoretic foundations to computational implementation*. Springer, New York, NY, USA, 2 edition, 2001.
- [34] C. P. Robert and G. Casella. *Monte Carlo statistical methods*. Springer, New York, NY, USA, 2 edition, 2004.
- [35] W. J. Rugh. *Linear system theory*. Prentice Hall, Englewood Cliffs, NJ, USA, 1993.
- [36] T. B. Schön, F. Lindsten, J. Dahlin, J. Wågberg, C. A. Naesseth, A. Svensson, and L. Dai. Sequential Monte Carlo methods for system identification. In *Proceedings of the 17<sup>th</sup> IFAC Symposium on System Identification (SYSID)*, pages 775–786, Beijing, China, Oct. 2015.
- [37] T. B. Schön, A. Svensson, L. M. Murray, and F. Lindsten. Probabilistic modelling of nonlinear dynamical systems—the use of sequential Monte Carlo. *Mechanical Systems and Signal Processing*, 2017. Under review for the same special issue.
- [38] J. Schoukens, J. Suykens, and L. Ljung. Wiener-hammerstein benchmark. In *Proceedings of the 15th IFAC Symposium on system identification (SYSID)*, St. Malo, France, July 2009.
- [39] M. Schoukens and J.-P. Noël. Wiener-Hammerstein benchmark with process noise. In *Workshop on Nonlinear System Identification Benchmarks*, pages 15–19, Brussels, Belgium, April 2016.
- [40] A. Wills, T. B. Schön, F. Lindsten, and B. Ninness. Estimation of linear systems using a Gibbs sampler. In *Proceedings of the 16<sup>th</sup> IFAC Symposium on System Identification (SYSID)*, pages 203–208, Brussels, Belgium, July 2012.